PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

Order N°: .....

UNIVERSITY OF M'SILA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Computing science Department

Dissertation

Submitted in fulfillment for the requirement of Bachelor's degree in Computing

Domain: Mathematics and computing
Branch: computing science
Specialty: fondamental computing

THEME

Transformation of XML document into a relational database

Presented by:
MAMMERI Asma
SEGHIRI Karima
BOUTCHICHA Nossiba

Supervised by :
Miss.Meliouh Amel
Dr.L.Belabdelouhab-Fernini

Promotion : 2012 /2013

Dedication

With an immensejoy and enormous pleasure I dedicate this work to my mother who helped me all my life and was an example of the great mother. To my father who supported me and encouraged me.

To my lovely sisters: Asma, Iman

To my cousins : Leila Bisker, Asma Bisker.

To all the members of Boutchicha and Benaziez families

To all my friends and to all the people that know me.

NOUSSEIBA

Dedication

Everyone who remembers his or her own educational experience remembers teachers, not methods and techniques, So firstly I dedicate this work to all my teachers from the primary school to this year.

To my mother and father who helped me

To my sisters and brothers for their support

To my nephews and nieces especially"Aicha"

Karima

Dedication

I dedicate this work

To my father who guided my first steps in life, may ALLAH protect him.

To my mother who informed me my way and who encouraged and supported me throughout my all studies.

To my brother AlaEddine.

To my sister Ouiem

To my beautiful family.

To all my friends who left me the best memories.

ASMA

Thanks

We would first like to thank ALLAH the Almighty and merciful, who gave us the strength and patience to accomplish this modest work.

Secondly, we wish to thank our Supervisors Miss.Meliouh Amel and Mrs. L. Belabdelouahab-Fernini, for their valuable advice and assistance throughout the period of work
Special thanks to the professors Roussafi Mahdjoubi, Rabah Mokhtari, Mohamed kamel and lalia Souadi, who gave us the hope, and help to work.

Finally, we would like to thank all those who participated in any way to help us succeed in this work.

Table of contents

LIST OF FIGURES

LIST OF Tables

GENERAL

INTRODUCTION

CHAPTER 1

TRANSFORMATION

CHAPTER 2

XML AND RELATIONAL DATABASES

CHAPTER 3

CONTRIBUTION

GENERAL

CONCLUSION

Model-driven engineering (MDE) is a software development methodology which focuses on creating and exploiting domain models (that is, abstract representations of the knowledge and activities that govern a particular application domain), rather than on the computing (or algorithmic) concepts . MDA (Model Driven Architecture) basic concepts are the key challenge of transforming models into other models or codes.

Transforming a model into another model means; a source model is transformed into a target model based on some transformation rules. Different methods can be used for defining those rules.

Transformation technique in computer science is very important because it defines a large domain, in which we can find diverse concepts, and facilities in programming and verifying applications.

In this context, we mention that in our work we develop a specific transformation which is code to model and to be more specific, the transformation is from xml code to relational database model.

A database is a means of storing information in such a way that information can be retrieved from it. In simplest terms, a relational database is one that presents information in tables with rows and columns. A table is referred as a relation in the sense that it is a collection of objects of the same type (rows). Data in a table can be related according to common keys or concepts, and the ability to retrieve related data from a table is the basis for the term relational database. There are many management systems and one of them MySQL; it’s the world's most widely used open source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases.

On the other hand XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is known by XML 1.0 version , produced by the W3C. It is fast emerging as the dominant standard for representing data in Internet, so there are increasing needs to store large volume of data.

Both XML documents and relational databases store data. Both have established techniques for extracting the data they contain ,but relational databases are better for handling large volumes of data within a system. They have mature management systems that can efficiently and reliably maintain large quantities of structured data. This data can be updated via transactions that ensure the integrity of the database and the content can be extracted very quickly when properly configured. There is no equivalent XML management system, and using XML documents directly for storing and maintaining large volumes of data can prove both inefficient and unreliable.

The Web and XML have influenced all walks of lives of those who transact business over the Internet. People like to do their transactions from their homes to save time and money. Most companies, including banks, maintain their records using relational database technology. But the traditional relational database technology is unable to provide all these new facilities to the customers. To make the traditional relational database technology cope with the Web and XML technologies we need a do transformation from an XML document into a relational database model, taking into consideration the requirement of relational database technology.

Many applications are developed in transformation domain, we can denote transformation of class diagram to XML document using ATOM3, automatic transformation of SWLR to Prolog using XSLT, automatic transformation of ontology's OWL to prolog and transformation of class diagram of UML to ontology's language OWL.

In this context, our objective is to transform an XML document to a relational database .To realize this transformation, we must load an XML (with DTD or simple) document, analyze it for syntaxes errors, then applied a transformation grammar that we must define; the grammar has the role of transforming an XML code into a relational database formed of certain number of tables. Finally tables must be relied by database techniques.

Our manuscript is organized in 3 chapters, the first one, talk about the transformation in the world of data processing and the architecture of MDA. The second is about the XML and relational databases in addition of similarities and differences between them. Third chapter presents the conception of our application and results obtained.

1 Introduction

In the world of data processing we find the words **MDE** and **MDA** which present the domain of realizing transformations from a formalism to another. This domain is large and more used actually in computer science .

In this chapter we will define the influence of the previous principles MDE and MDA in the transformation by suiting definitions and functionalities for each of them.

2 Definitions

2.1 Model-driven engineering (MDE): is a software development methodology which focuses on creating models. Its purpose is to increase productivity by maximizing compatibility between systems, simplifying the process of design, and promoting communication between individuals and teams operating on the system.

In our work we adopt the Model Driven Architecture (MDA) introduced by the Object Management Group (OMG)[1] . [4]

2.2 Model Driven Architecture (MDA):A specific approach defined by the OMG toward model driven development (MDD) that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform. The focus of MDA is primarily the provision of standard specifications for useful technologies in the MDD space. Examples are the UML (modeling), MOF (meta-modeling),transformations (QVT), platform specific annotations (set of UML profiles) [7].

2.3 Transformation: a set of rules describing the conversion of one syntactic structure into another related syntactic structure . [12]

2.4 Model: A model is a set of statements defining an abstraction of a system (other problem addressed by that system) under study and fulfilling a particular purpose.

Models have been used for ages in various scientific disciplines including biology, economy, House building or geography in which the use of maps is common practice.

[1] Object Management Group is an American association non-profit organization founded in 1989 whose purpose is to standardize and promote the subject in all its forms model.

The set of statements expressed in a model typically uses a set of predefined modeling constructs and constraints restricting statement usage. It is the role of a meta-model to define these constructs (i.e. their abstract syntax) and constraints; without a metamodel, models cannot be built so that they can be validated or processed by tools . [6]

2.5 Meta-model: A meta-model is a model whose modeled system/problem under study is a set of models and whose purpose is the definition of their abstract syntax.

2.6 Transformation operation :A transformation operation consists in describing how one or more constructs in the source language can be transformed into one or more constructs in the target language.[6]

2.7 Transformation definition: A transformation definition is a set of transformation operations, assembled together in accordance with a particular transformation language, which describe how one or more source models are transformed into one or more target models.[6]

2.8 Transformation language: A transformation language is a modeling language for transformation definitions.[6]

2.9 Model Transformation : The process of converting one model to another model of the same system. Allowing several models as input or output and define model transformation as the "automatic generation of one or multiple target models from one or multiple source models, according to a transformation description". This is the definition we will use in this document. [5]

2.10 Graph Transformation: Graph transformation languages build on theoretical foundations of algebraic graph grammars and are a subcategory of declarative languages. Graph transformations have interesting theoretical properties and are often used in formal approaches and proofs. Models are interpreted as graphs, and graph transformations manipulate sub-graphs.

Triple Graph Grammars (TGG) are a way of describing graph transformations.

They have rules that are specified by three graphs:

- Left-hand side graph: sub-graph of the source graph
- Right-hand side graph: sub-graph of the target graph
- Correspondence graph: describes the mapping between elements of the Left-hand side graph and elements of the right-hand side graph.

The left-hand side describes the precondition for the application of the rule, the Right-hand side describes the post-condition of the rule. [10]

2 Architecture levels of MDA

The architecture of MDA is organized into four levels

- **M0 level:** corresponds to the real world. This represents the real information of the user.
- **M1 level:** consists of information models, which allow the description of information M0. Any model is expressed in one language, whose definition is explicitly provided in M2 level.
- **M2 level:** is composed of meta-models. A metamodel can be defined as a model of a modeling language; which serves better express the common concepts of all models of a domain.
- **M3 level:** consists of the MOF, also called metameta-model which is the language for defining meta-models .[5]

3 Types of Transformations

4.1 Vertical transformations / Horizontal transformations / Oblique transformations

4.1.1 Horizontal transformations

Horizontal transformations may be refactoring or delocalizing transformations. Refactoring transformations reorganize a specification to improve its design without changing its meaning. Refactoring can be applied to both tree and graph-based languages. Delocalizing transformations are used for optimization, or to compose parts of the same implementation that are specified independently. General purpose programming language compilers use horizontal transformations to optimize generated binary code to improve performance, and aspect weaving uses them to compose independently specified aspects. Like refactoring, aspect weaving can be applied to both tree and graph-based languages. [8]

4.1.2 Vertical transformations

Vertical transformations are mostly refinement transformations that map higher level models to lower level models, such as SQL DDL files, to general purpose programming language source code, or to other low level development artifacts. [8]

4.1.3 Oblique transformations

Oblique transformations combine horizontal and vertical transformations. General purpose programming language compilers generally perform oblique transformations, combining a vertical transformation that generates binary code from the source code, and a horizontal transformation that optimizes the generated binary code. [8]

4.2 Endogenous transformations/ Exogenous transformations

Endogenous transformations are transformations between models expressed in the same language. Exogenous transformations are transformations between models expressed using different languages .[12]

4.3 Transformation PIM, PSM ,CIM,PDM

4.3.1 Platform Independent Model (PIM)

The PIM describes the system regardless of target platform on which it will run. It therefore presents a detailed functional view of the system, without technical details. [9]

4.3.2 The Computation Independent Model (CIM)

Also called domain model or business model, the CIM capture requirements in terms of needs and describes the situation in which the system will be used. Its purpose is to assist in the understanding of the problem but also to establish a common vocabulary for a particular domain. In practice, the term "CIM" is used very little. [9]

4.3.3 The Platform Description Model(PDM)

The PDM model which describes a platform execution. It provides a set of technical concepts representing different parts of the platform and / or the services it provides. [9]

4.3.4 Platform Specific Model (PSM)

is the result of the combination of PIM and PDM. It represents a detailed technical overview of the system. It may exist with different levels of detail. In its most comprehensive form, it is the basis for the generation of the implementation. [9]

4.3.5 CIM to PIM : named requirement model to application model transformation.[6]

4.3.6 PIM to PIM : application model to application model transformation or application execution platform-allocation model to allocated model.[6]

4.3.7 PIM to PSM : named allocated model to implementation model transformation.[6]

4.3.8 PSM to PSM : implementation model to implementation model transformation.[6]

4.3.9 PSM to Code : implementation model to Java transformation.[6]

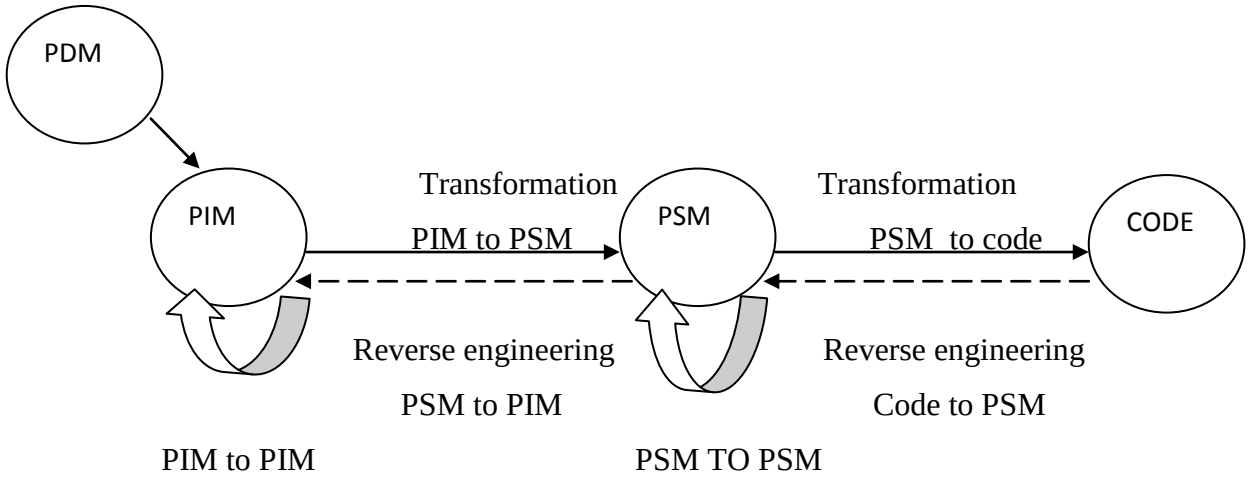

Figure 1.1 Models and Transformations in MDA Approaches

4.4 Transformation model to model / model to code

4.4.1 Model to-code Approaches

In this category, there are two approaches :

- Transformations based on the principle of visitor:

involves crossing the model by adding elements that reduce the semantic difference between the model and the target programming language.

The code is obtained by traversing the rich model to create a text stream. [9]

- Transformations based on the principle of employers:

currently are the more common. Target code contains pieces of meta-code used to access the information source model. [9]

4.4.2 Model to-Model Approaches

There are transformations based on direct manipulation: these changes are based on an internal representation of source and target models, and a set of APIs, an API to manipulate the internal representation of models. [9]

5 Model Transformation Languages, Tools and Standards

In this section we briefly introduce different model transformation languages and tools:

5.1 EMF Henshin

Is a continuation of the EMF Tiger [2] transformation language. It is an in-place model-to-model transformation language using triple graph grammars (TGG). It is based on the Eclipse Modeling Framework EMF .The transformation description is a transformation model

[2] Eclipse model transformation tool.

consisting of a left-hand-side graph, a right-hand-side and a list of correspondence mappings. The graph nodes are model element instances of the source metamodel and the target metamodel, respectively. It is thus possible to create higher-order transformations with EMF Henshin. There is no built-in support for creating traces, no support for multi-directionality or incremental model transformation. [10]

5.2 ATL

The ATLAS Transformation Language (ATL) is a hybrid model-to-model transformation language. ATL supports both declarative and imperative constructs.

The preferred style is declarative, which allows a cleaner and simpler implementation for simple mappings. However, imperative constructs are provided so that some mappings that are too complex to be handled declaratively can still be specified. An ATL transformation program is composed of rules that describe how to create and initialize the elements of the target models. The language is specified both as a metamodel and as a textual concrete syntax.

ATL is integrated in the Eclipse development environment and can handle models based on EMF. ATL also provides support for models using EMF-based UML profiles. ATL-code is compiled and then executed by the ATL transformation engine. ATL supports only unidirectional transformations. ATL offers dedicated support for tracing. The order of the rule execution is determined automatically, with the exception of lazy rules, which need to be called explicitly.

Helper functions provide imperative constructs. ATL does not support incremental model transformation, so a complete source model is read and complete target model is created. Manual changes in the target model are not preserved. ATL supports a mode for in-place transformation, called the “refining mode”. It has limitations and cannot be used in combination with certain constructs, e.g. with lazy rules. [10]

5.3 Query/View/Transformation (QVT)

Query/View/Transformation (QVT) is a standardized language for model transformation established by the Object Management Group (OMG). QVT uses the Object Constraint Language (OCL), Meta Object Facility (MOF) and is aligned with the Model Driven Architecture (MDA) QVT defines three languages for model-to-model transformations. QVT defines both a textual concrete syntax and a XMI-based metamodel for creating model representations of QVT transformations. QVT has a black box mechanism that allows calling external code from within the transformation. For the tool implementation of each of the

three languages QVT defines four conformance classes for interoperability: syntax executable (ability to execute QVT in the concrete syntax), XMI executable (ability to execute QVT in a serialized XMI model), syntax exportable (ability to export QVT into the concrete syntax), XMI exportable (ability to export QVT into a serialized XMI model).QVT defines three transformation languages:

- QVT Relational is a high-level declarative transformation language. Both a graphical and a textual syntax are defined for QVT. The language supports the specification of bidirectional transformations.

  When a bidirectional transformation is executed, the execution direction needs to be specified. A transformation is specified as a set of relations between the source and target metamodel that must hold true.

  This transformation can be used to check two models for consistency, to enforce consistency by modifying the target model, to synchronize two models and for in-place transformations. It supports complex pattern matching using OCL. Trace models are created implicitly. The semantics is defined by a mapping to QVT Core.
- QVT Core is a simple, low-level declarative model transformation language. It serves as a foundation for QVT Relational and is equally expressive. It supports pattern matching over a flat set of variables, where the variables of source, target and trace models are treated symmetrically. Trace models must be defined explicitly.
- QVT Operational is an imperative model transformation language that extends QVT Relational with imperative constructs. The transformations are unidirectional. It uses implicit trace models. [10]

5.4 Smart QVT

Smart QVT is an implementation of a transformation engine for Operational QVT. It is an imperative language for model-to-model transformation for EMF based models. The transformation description is compiled into Java code and supports the QVT black box mechanism to call external code. It offers built-in tracing support. In addition it offers reflection to access tracing information, such as the target object corresponding to a source object or the source object corresponding to a target object. There is support for control parameters and higher order rules. Incremental model transformation is not supported, as the complete source model is read and the complete target model is created. There is currently no support for multi-directionality. [10]

5.5 ModelMorf

ModelMorf is an implementation of the OMG standard QVT Relational. It is a declarative model-to-model transformation. It supports multi-directionality, so the same rule can be used to map in both directions. It supports target incremental model transformation (called change propagation semantics), so if the source model changes only the changed part of the model is transformed. It provides built-in functionality to create traces. It is possible to create in-place transformations. Furthermore, it is possible to compose model transformations to build and extend complex transformations from simpler ones. There is currently no support for aspect-orientation, reflexion or source incremental model transformation. [10]

5.6 Open Architecture Ware (OAW)

OAW integrates a number of tools for model transformations into a coherent framework. OAW provides a workflow specification language and the transformation language Xpand. The workflow language is used to control the transformation process and to specify the sequence of transformations between the different models. The Xpand transformationlanguage is a template-based, imperative language for model-to-text transformations. OAW is distributed as a plugin of the Eclipse platform and is able to handle EMF models. [10]

5.7 Kermeta

Kermeta is a general purpose modeling and imperative programming language, also able to perform transformations. It offers EMF-based meta-modeling, constraints, checks, transformation and behavior support. Models and meta-models need to be explicitly loaded and stored. Target elements need to be explicitly instantiated and added to the target model, which requires more code. Rule application control and rule scheduling needs to be specified explicitly by the user.

Kermeta supports reflection, exception handling and aspect-orientation. There is no built-in support for traceability and multi-directionality. Incremental model transformation is not supported, so the complete source model is read and complete target model is created when the transformation is executed. [10]

5.8 ETL

The Epsilon Transformation Language (ETL) is a hybrid model-to-model transformation language. It is part of the Epsilon model management infrastructure. It can handle several source and several target models. It offers rule scheduling functionality: lazy rules are only

executed, when they are explicitly called, guarded rules are only executed if their guard evaluates to true, greedy rules are executed whenever possible. Rules can be reused and extended through rule inheritance. External code can be executed from within the transformation rule. [10]

5.9 XML Stylesheet Language Transformations (XSLT)

XSLT is a functional transformation language for manipulating XML data.Being a functional language, rules have to be called explicitly. There is no built-in traceability support and rules are strictly unidirectional. Transformations are stateful, so there is no support for incremental transformation. XSLT transformation descriptions are themselves XML documents, so higher-order transformations can be realized. Due to the fact the XSLT was initially developed to transform XML documents into HTML documents, XSLT is limited to simple transformations . [10]

5.10 More...

Many more transformation languages exist. We list some names and references here, without claiming to be complete: MofLog, GreAT,GenGen, Beanbag, UML, UMLX, ATOM, VIATRA, BOTL,XDE Transformations,Coadagen Architect transformation, b+mGenerator Framework,OptimaJTransformations,ArcStylerTransformations,MPSTransformations,MicrosoftDSLToolTransformations,Metaedit+Transformations,AndroMDA,JET,FUUT-JE,GMT,Jamda,Fujba Transformations, TXL,Stratego.[10]

6 Conclusion

The transformation is too broad domain to describe in few pages. So we talked about the transformation in general like transformation model to model, model to code and reverse in data processing .We found many transformation’s applications , for example xml to relational data base ,on that focus our work.

1 Introduction

Databases have been a staple of business computing from the very beginning of the digital area. In fact, the relational database was born in 1970, created using a special computer language, Structured Query Language (SQL) that is the standard for database interoperability. SQL is the foundation for all of the popular database applications available today, from Access to Oracle. We talk about XML Short for e**X**tensible **M**arkup **L**anguage, a specification developed by the W3C. XML is a pared-down version of SGML, designed especially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

This chapter talks about XML, and the relational databases in more details.

2 XML

2.1 History

XML emerged as a way to overcome the shortcomings of its two predecessors, SGML and HTML which were both very successful markup languages, but which were both flawed in certain ways.

SGML, the international standard for marking up data, has been used since the 80s. SGML is an extremely powerful and extensible tool for semantic markup which is particularly useful for cataloging and indexing data. Like XML, SGML can be used to create an infinite number of markup languages and has a host of other resources as well.

However, SGML is pretty complex, especially for the everyday uses of the web. Not only that, but SGML is pretty expensive. Adding SGML capability to a word processor could double or triple the price. Finally, the commercial browsers made it pretty clear that they did not intend to ever support SGML.

HTML on the other hand was free, simple and widely supported. HTML was originally designed at CERN around 1990 to provide a very simple version of SGML which could be used by "regular" people. As everyone knows, HTML spread like wildfire.

Unfortunately, HTML had serious defects, HTML is static. It's not a programming language; it's a markup language, so you can't do things like save user input, let users log in, etc. HTML is weakened because of its complexity and incompatibility.

So in 1996, discussions began which focused on how to define a markup language with the power and extensibility of SGML but with the simplicity of HTML. The World Wide Web

Consortium (W3C) decided to sponsor a group of SGML gurus including Jon Bosak [1] from Sun.

Essentially, Bosak and his team did to SGML what the Java team had done to C++. All of the non-essential, unused, cryptic parts of SGML were sliced away. What remained was a lean, mean marking up machine: XML. The specification of XML (written mostly by Tim Bray [2] and C.M. Sperberg-McQueen [3]) was only 26 pages as opposed to the 500+ pages of the SGML specification! Nevertheless, all the useful things which could be done by SGML, could also be done with XML.

Over the next few years, XML evolved, drawing from the work of its sponsors and the work of developers solving similar problems such as Peter Murray Rust [4] who had been working on CML (Chemical Markup Language) and the consortium of folks working on MathML. By mid 1997 the eXtensible Linking Language XLL project was underway and by the summer of 1997, Microsoft had launched the Channel Definition Format (CDF) as one of the first real-world applications of XML.

Finally, in 1998, the W3C approved Version 1.0 of the XML specification and a new language was born. [20]

2.2 Understanding XML

XML is a standardized method of describing information, or data, established by the W3C. It is a tag based language, similar to HTML, but using stricter standards. Unlike HTML, tags are not predefined, they can be established based on the user's needs at design time.

[1] Jon Bosak led the creation of the XML specification at the W3C. From 1996–2008, he worked for Sun Microsystems.

[2] Timothy William Bray (born June 21, 1955) is a Canadian software developer and entrepreneur.. Bray was also one of the main authors of the original XML specification. Bray was Director of Web Technologies at Sun Microsystems from early 2004 to early 2010.

[3] C. M. "Michael" Sperberg-McQueen is an American markup specialist. He was co-editor of the Extensible Markup Language (XML) 1.0 spec (1998), and chair of the XML Schema working group.

[4] Peter Murray-Rust is a chemist specializing in informatics, born in Guildford in 1941.

2.3 XML syntax

2.3.1 Elements and Texts inside elements

Markup Elements enclosed in Tags

<element>…</element>

<element/> (empty element tag)

Elements tags can be nested and mixed with text

<element>…

<subElement>…</subElement>…

</element>

Warning: make sure nesting is well formed!

</element>_{<element>}

2.3.2 Attributes

Element attributes

<element attribute="…">…</element>

<element attribute="…"/>

Do we really need attributes?

<element>

<attribute>…</attribute>

</element>

Attributes represent "hidden" data about the element, while element text is "visible"

Warning: Attribute names must be unique!

<element a="1" a="ABC"></element>

2.3.3 Namespace Declarations (special attributes)

Should whitespace (spc, tabs, cr, lf) be ignored?

It depends:

– Data oriented XML ignore whitespace (more compact documents, faster processing, less readability).

– Document oriented XML whitespace is part of the content and must be preserved.

2.3.4 Prolog

Header found at the beginning of all XML Documents

<?xml version="1.0" encoding="UTF-8"?>.

– Specify the encoding usually Unicode (UTF-8, UTF-16) or ISO- 8859-1.

2.3.5 Entities (&) Character Data (CDATA)

Special characters can be encoded using character entities.

In general, any Unicode character can be entered with the notation:

&#N; (N is decimal)

&#xM; (M is hex)

Character	Entity
<	&lt ;
>	&gt ;
&	&amp ;
"	&quot ;
‘	&apos ;

Table 2.1 Encoding of special characters

Character **Data** Sections are used to stop parsing the XML markup and treat the enclosed data as is:

<element>

<![CDATA[<not An Element>]]>

</element>

2.3.6 Comments

Comments may appear anywhere in a document outside other markup. Comments cannot appear before the XML declaration. Comments start with "<!--" and end with "-->". The string "--" (double-hyphen) is not allowed inside comments; this means comments cannot be nested. The ampersand has no special significance within comments, so entity and character references are not recognized as such, and there is no way to represent characters outside the character set of the document encoding.

An example of a valid comment:

"<!-- no need to escape <code> & such in comments -->".[3]

2.4 XML Structure

2.4.1 Document Type Definition

A **D**ocument **T**ype **D**efinition (DTD) is an optional part of an XML document that defines the document’s exact layout and structure. Although not essential, there are several advantages to using a DTD, and these advantages become more obvious, and of greater value, as the size and complexity of the XML increases. Most non-trivial applications of XML

benefit from a DTD, so most document authors need to know how to write them, and most software developers working with XML need to know how to take advantage of them.

Among the more important things that a DTD describes are:

- The elements that can appear in a document.
- The order in which they can appear.
- The ways some elements can be contained within instances of others.
- Which elements are optional and which are mandatory.
- Which elements attributes apply to.
- Whether attributes they are mandatory or optional.
- Whether attributes can have default values.
- Where free text can be placed in the document . [21]

a) Internal DTD Declaration

If the DTD is declared inside the XML file, it should be wrapped in a DOCTYPE definition with the following syntax :

<!DOCTYPE root-element [element-declarations]>. [18]

b) External DTD Declaration

If the DTD is declared in an external file, it should be wrapped in a DOCTYPE definition with the following syntax:

<!DOCTYPE root-element SYSTEM "filename">. [18]

c) The syntax of the DTD

- CDATA keyword in an attribute declaration, has a different meaning from the CDATA section in an XML document. In CDATA section all characters are legal (including <,>,&, ' , ").
- PCDATA keyword specifies that the element must contain parsable character data – that is, any text except the characters less-than (<) , greater-than (>) , ampersand (&), quote(') and double quote (") .
- #PCDATA :means when we define element-content as #PCDATA: <!ELMENT name (#PCDATA)> ;the type of the element is string .
- CDATA means when we define attribute type as CDATA :the type of the attribute is string .
- #IMPLIED: Specifies that there is no default value for this attribute, and that the attribute is optional.

- #REQUIRED : There is no default value for this attribute ,but a value must be assigned.
- #FIXED: it specifies that the value must be the value provided. [19]

The Example below gives more explication:

```
<!DOCTYPE family[
<!ELEMENT family(father)>
<!ELEMENT father(#PCDATA)>
<!ATTLIST father idf ID #REQUIRED>
<!ATTLIST father adresse CDATA #FIXED>
<!ATTLIST father jobe CDATA #IMPLIED>
]>
```

d) What is a standard DTD (DTDs)

Example of the standard DTD format

```
<!ELEMENT Library (Books+, Publishers+)>
<!ELEMENT Books (#PCDATA)>
<!ATTLIST Books LCNo ID #REQUIRED
                PName IDREF #REQUIRED
                title CDATA #FIXED
                author CDATA #FIXED>
<!ELEMENT Publishers (#PCDATA)>
<!ATTLIST Publishers PName ID #REQUIRED
                PAddr CDATA #FIXED
                PCity CDATA #FIXED>.
```

Usually the attributes pertaining to a given element are declared in a single expression, but it is just a convention adopted by a lot of DTD writers. [11]

e) Why using a standard DTD?

- Expedites interchange to use a known model .
- users are already familiar with it .
- Some tools are optimized for certain standards .
- A standard may be mandated in a given industry. [1]

2.4.2 W3C XML Schema: (XML Schema)

The W3C XML Schema Definition Language commonly referred to as W3C XML Schema, or even just XML Schema, is one of several languages that provide additional definition constructs for data modeling. The resulting data model is called a schema, and it can define the same element and attribute structures a DTD can define and a great deal more. Additional capabilities include more specific hierarchical rules than a DTD, several data modeling techniques for easing definition and maintenance of a schema, as well as a powerful set of constructs for defining data typing rules for element and attribute content. [2]

These advanced capabilities make XML schemas highly suitable for business system and database applications that were inordinately constrained by the limitations of DTDs. It is useful to mention that there are several alternative XML schema languages in development. [2]

When the data used in a business system need to be maintained in a form that is more complex than can be expressed in DTD rules, XML Schema can be used to enforce these rules and ensure that the data meets business processing needs. For example, an organization that creates, stores and manages product specification information (spec sheets) would want to make sure that certain fields of information are maintained in a specific format. This enables the data to move seamlessly between applications and databases. [2]

2.4.3 Disadvantages of Schema

- Multiple standard.
- Does not support entities.
- Not supported by all validating parsers **.**[17]

2.4.4 Advantages of DTD

- Have a simpler syntax.
- Require fewer lines of code to be created.
- Are more widely accepted by tools and browsers that work with XML.
- Has one form standard.
- Supports entities.
- Supported by all validating parsers. [17]

2.4.5 Well Formed XML Documents

A "Well Formed" XML document has correct XML syntax. The syntax rules were described in following points:

- XML documents must have a root element.
- XML elements must have a closing tag.
- XML tags are case sensitive.
- XML elements must be properly nested.
- XML attribute values must be quoted. [16]

2.4.6 Valid XML Documents

A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a Document Type Definition (DTD). [16]

3 Relational Databases

A relational database consists of a collection of tables that store particular sets of data. The invention of this database system has standardized the way that data is stored and processed. The concept of a relational database derives from the principles of relational algebra, realized as a whole by the father of relational databases, E. F. Codd [5] .

Most of the database systems in use today are based on the relational system, however relational databases are built-in to software that people and companies purchase, so the database is generally invisible to the general public. [14]

3.1 Database Content

The relation, which is a two-dimensional table, is the primary unit of storage in a relational database. A relational database can contain one or more of these tables, each table consisting of a unique set of rows and columns. A single record is stored in a table as a row, also known as a tuple, while attributes of the data are defined in columns, or fields, in the table. The characteristics of the data, or the column, relates one record to another. Each column has a unique name and the content within it must be of the same type. [14]

[5] Edgar Frank "Ted" Codd (August 23, 1923 – April 18, 2003) was an English computer scientist who, while working for IBM, invented the relational model for database management, the theoretical basis for relational databases.

3.2 Organizing Data

The data that is stored in tables is organized in a logical manner based on a particular purpose to help minimize duplication, reduce data anomalies, and reinforce data integrity. The process by which data is logically organized is called normalization; it not only simplifies the way data is defined, but it also regulates its structure. There are five forms in the normalization process, with each form meeting a more demanding condition. [14]

3.3 Database Management System (DBMS)

Stored data is manipulated using a programming language Many varieties exist, but most are based on set theory relational operators such as *and*, *or*, *not*, and *in*, all of which are used to perform operations on the data. The operations that can be used in relational databases include *insert*, *select*, *update*, and *delete* privileges the most famous of this management systems are:Oracle Database ,Microsoft SQL Server ,MySQL (Oracle Corporation),IBM DB2 ,IBM Informix ,SAP Sybase Adaptive Server Enterprise ,SAP Sybase IQ and Teradata. [14]

4 Relation between XML and Database

XML and databases are similar in that they are both structured methods of organizing data. Each can contain a key value associated with one or more instances of data items. In XML, this would be multiple tags with the same name , such as : <product>item1</product><product>item 2</product> . In a database, this is represented by a unique value of same type. [15]

XML data is hierarchical, however relational data is represented in a model of logical relationships: an XML document contains information about the relationship of data items to each other in hierarchical form but in relational model, the only types of relationships that can be defined are parent table and dependent table relationships. [15]

XML data is self-describing**;** An XML document contains not only the data, but also tagging for the data that explains what it is. A single document can have different types of data. With the relational model, the content of the data is defined by its column definition. All data in a column must have the same type of data. [15]

XML data has inherent ordering**;** the order in which data items are specified is assumed to be the order of the data in the document. There is often no other way to specify order within the document. For relational data, the order of the rows is not guaranteed unless you specify an ORDER BY clause on one or more columns. [15]

5 Conclusion

Relational databases have been used for over 30 years. Relational databases are good at providing a database that is simple, robust, flexible, scalable, and compatible. The XML was born in 1998, it takes a great part in the data possessing however XML databases run slower besides that XML has slower querying and searching functionality than other databases. The searches must sort through the text based information as well as the tags, which is slower than a search of only cell contents in a relational database.

1 Introduction

We talked in the first and second chapter ,about the transformation, XML and relational databases, so in this chapter we will talk about the transformation from XML document to a relational database and give an idea about our program which undertook this transformation respecting some rules .We define them to make sure that the target model will be more concrete and which is our purpose.

2 The proposed approach

We use in this transformation the **NetBeans** to create class in java and **MySQL Workbench** to create and design a data base generated by the analyzing of the XML document.

2.1 Java

Java is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible.

2.2 NetBeans

NetBeans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications.

2.3 IDE

An integrated development environment (IDE) or interactive development environment is a software application that provides comprehensive facilities to computer for software development. An IDE normally consists of a source code editor, automation tools and a debugger. Several modern IDEs integrate with Intel-sense coding features.

2.4 Relational database

It is a database that has a collection of tables of data items, all of which is formally described and organized according to the relational model. The term is in contrast to only one table as the database, and in contrast to other models which also have many tables in one database.

In the relational model, each table schema must identify a primary column used for identifying a row called the primary key. Tables can relate by using a foreign key that points to the primary key. The relational model offers various levels of refinement of the table

relations called normalization. The database management system (DBMS) or a relational database is called an RDBMS, and is the software of a relational database.

2.5 SQL (Structured Query Language)

It is a special-purpose programming language designed for managing data held in a relational database management systems.

2.6 MySQL Workbench

is a visual database design tool that integrates SQL development, administration, database, creation and maintenance into a single integrated development environment for the MySQL database system.

3 The general schema of the work

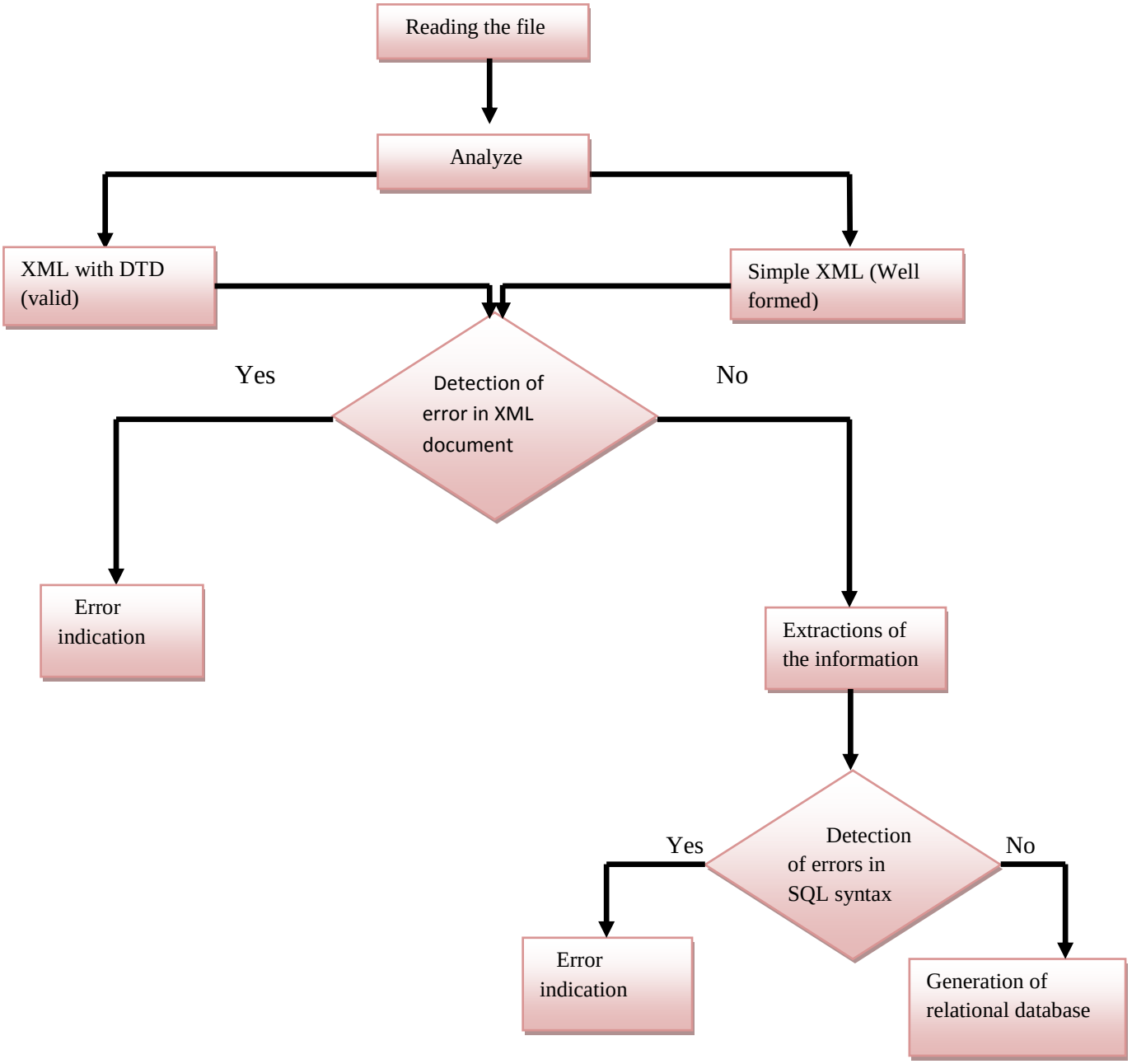

Figure 3.3.1 The general schema of the work

4 The general algorithm of the transformation

Begin

- Reading a XML document
- Analyzing the document

For the both type (valid, well formed) **do**

Detection of XML syntax errors

If (there is an error) **then**

- Canceling extraction
- Showing message of error

Else

- Extraction of XML information
- Detection of SQL's errors

If (error) **then**

- Canceling transformation
- Showing message of error

Else

- Application the transformation's grammar
- Generation of SQL tables
- Establishment of relations between tables
- Showing a message of the transformation's success

End if

End if

End for

End

4.1 Syntaxes errors detection

First of all, we check the prolog's syntax; it must contains the keyword "<?xml version="1.0" encoding="utf-8" standalone=yes?> » ;" standalone=yes" allows us to use interne DTD in addition of the keywords of the DTD (<!DOCTYPE, <!ELEMENT, <!ATTLIST, >,]>…), after that all the elements in the next block must be in the DTD block Considering the below symbols(<…>,</…>).

The only difference between the valid xml and the simple XML that we don't need to handle the block of DTD without ignoring this prolog's syntax <?xml version="1.0" encoding="utf-8"?> .

4.2 Extraction of information from the XML document

If there are no syntaxes errors in the XML document, the next step consists of extracting the following information :

- The root
- The elements
- The contents

The root is the first word after the keyword ("<! DOCTYPE"), the elements and the contents have the same manner of extraction. We mean that we extract each one which we find between tags , respecting some rules .In the Simple XML the root is the first tags after the prolog , however the rest of rules are the same for the elements and contents.

4.3 SQL errors detection

If a XML document has no errors, a second step is about SQL syntax verification (an example of errors is the use of some SQL's keywords in the contents of a XML document such as "use -, +… or affected a char to an INT or duplicate a primary key…..").

5 Transformation's grammar: Generation of SQL tables

After reading XML document and distinguishing the valid and the well-formed one, we have to apply a transformation grammar containing in 12 rules.

5.1 Grammar's rules

	Left hand side	Right hand side
R1	the word after ("<!DOCTYPE")	the schema's name
R2	the word after ("<!ELEMENT")	the table's name
R3	the word after ("<!ATTLIST")	the attribute of the table
R4	the word ID	primary key
R5	the word IDREF	forging key
R6	word CDATA	String
R7	the word #REQUIRED	obligatory attribute
R8	the word #IMPLIED	optional attribute
R9	the word #FIXED	constant attribute
R10	the symbols]>	the end DTD's block and the beginning of the content block
R11	"<>…. </>"	the extraction of the information between this two tags
R12	if (attribute==primary key)	attribute is a forging key

Table 3.1 Rules of transformation

The previous rules used for XML with DTD however the rule (R11) is used in case of a simple XML document.

6 Application's result:

The results obtained are shown in the followings figures. We give an example of XML document and SQL database.

- The first window consists of application interface, a click on it will take you to principal program.

Figure 3.6.1 Application's interface

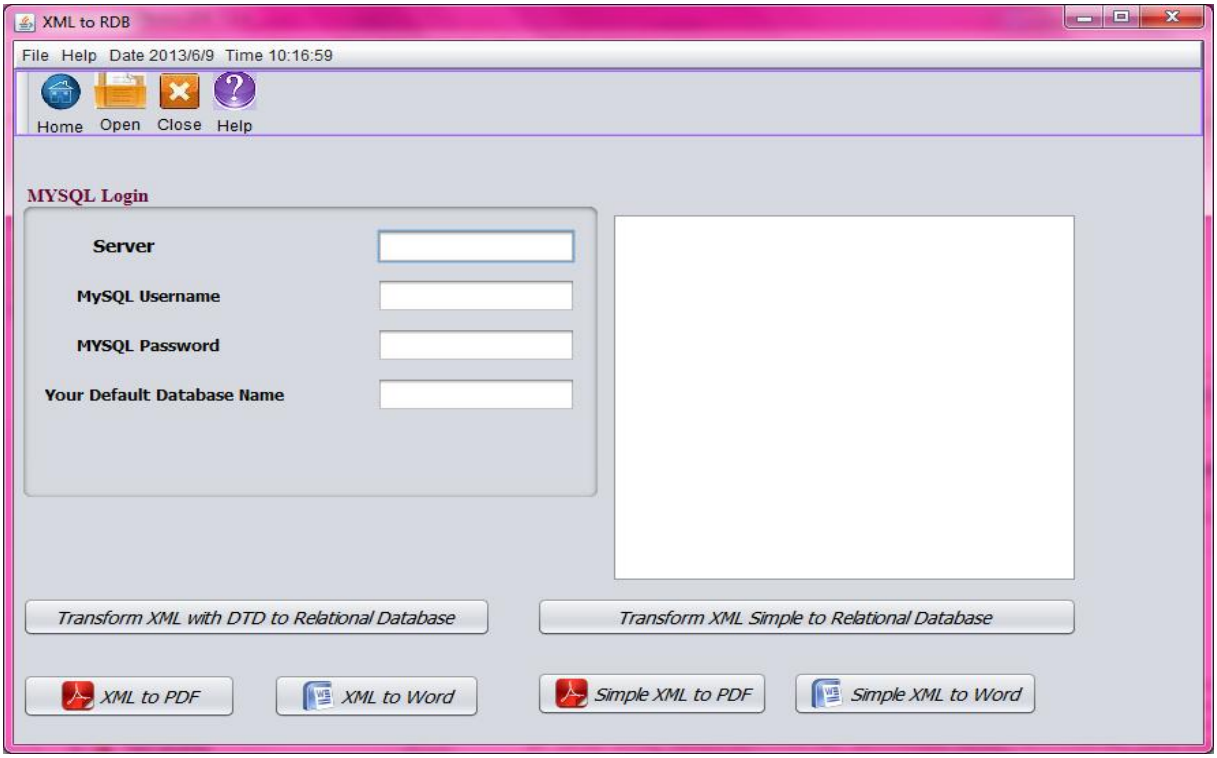

Figure 3.6.2 The principal window of the application

- This window contains the following :
  - **Menu Bar** that contains the following menus:
    - **File :** you find two Menu items : **Openfile** and **Exit** , **Openfile** (CTRL+O) allowed you to open a XML document to be transformed. **Exit** (CTRL+F4) cancel the current operation and close the application
    - **Help:** take you to the web site which give more information about the extensible markup language.
    - **Date** and **Time :** display the current date and time.
  - **Toolbar** contains four buttons **Home,Open**, **Close** and **Help :**
    - **Home:** allow you to initialize your information.
    - **Open:** take you to another window (**Figure 3.6.3)** that allow you to choose the file which will be transformed to a relational database.
    - **Close :** exit from the application.
    - **Help :** it is about how you can use the application.

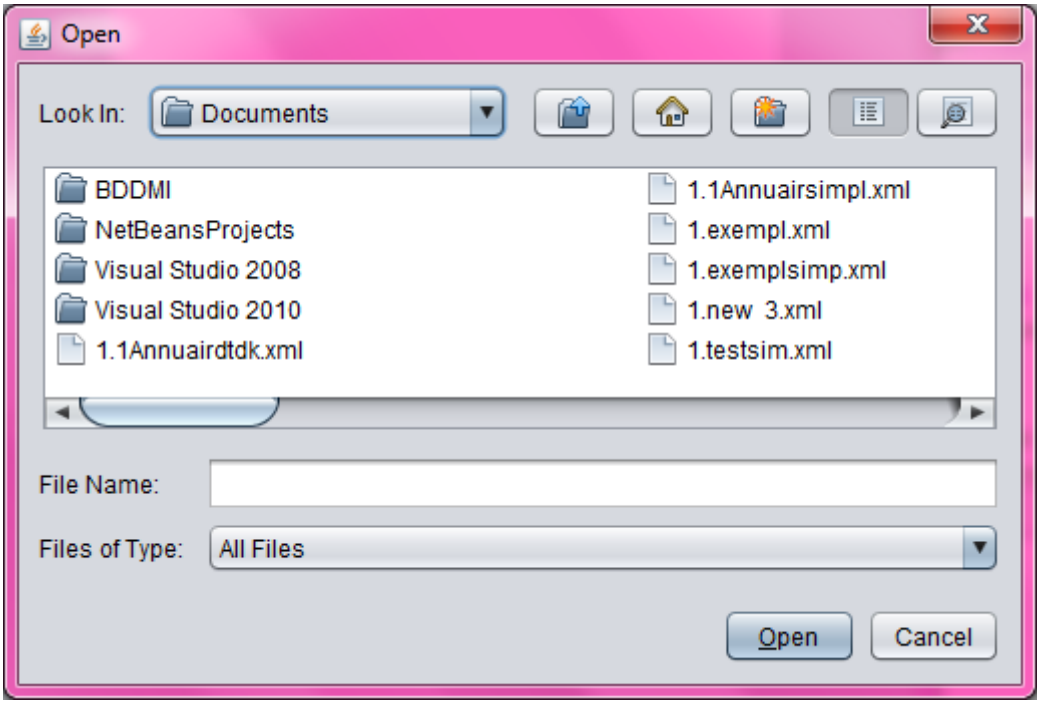

Figure 3.6.3 Choosing an XML document .

❖ After opening the XML document, you must choose between XML document with DTD or simple XML document, in **Figure 3.6.4** we have chosen XML with DTD.

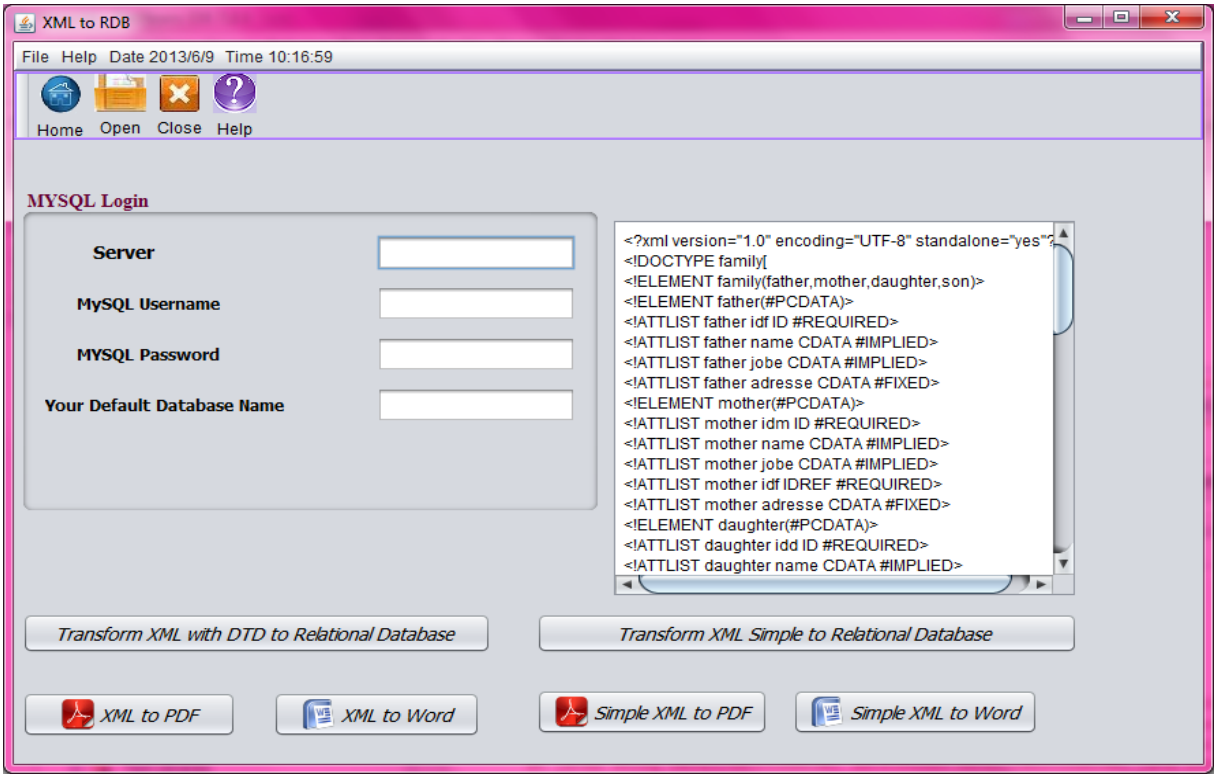

Figure 3.6.4 Appearance of XML document in the principal window

- if you want to transform your XML document(Simple or with DTD) to WORD or PDF Format ,choose one from the following buttons‹‹ XML to PDF››,‹‹XML to Word ››,‹‹Simple XML to PDF ››or‹‹Simple XML to Word ››. If your XML document is valid, a WORD or PDF document will be generated (it depends to your choice) Otherwise, an error message will appear.
- Next step is about entering information of MYSQL login(server name, Username, Password and Default Database Name) ,and clicking on the button ‹‹Transform XML with DTD to relational database››. If there are errors in XML syntax, the window of **Figure 3.6.5** will appear.

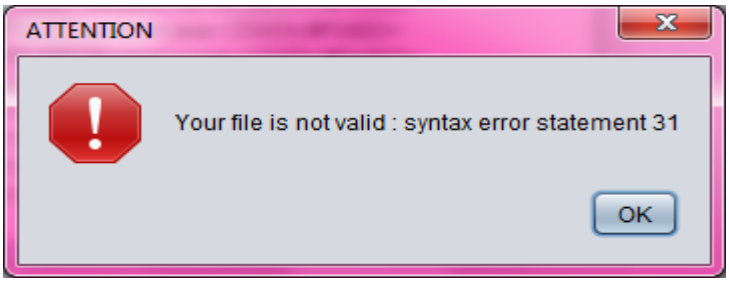

Figure 3.6.5 Syntax error message

Otherwise, another verification about SQL syntax will proceed, if there are errors you will get an error message**.**

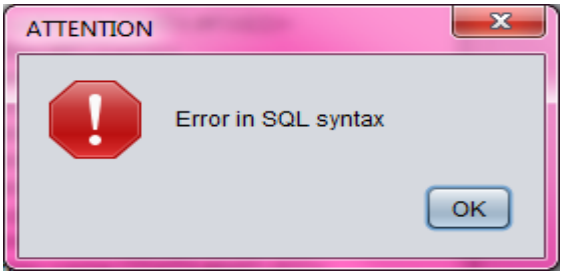

Figure 3.6.6 SQL error message

In contract case the transformation will be done successfully.

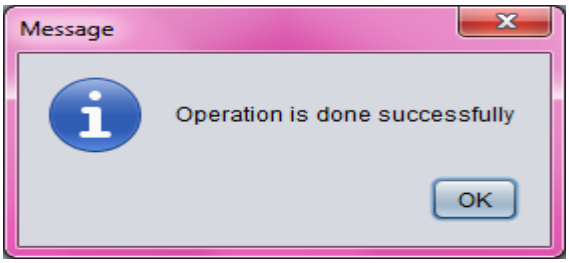

Figure 3.6.7 End of transformation process message

- The last window contains tables generated from the XML document, . To make sure that you have got your valid target , you can check MySQL server or MySQLWorkbench.

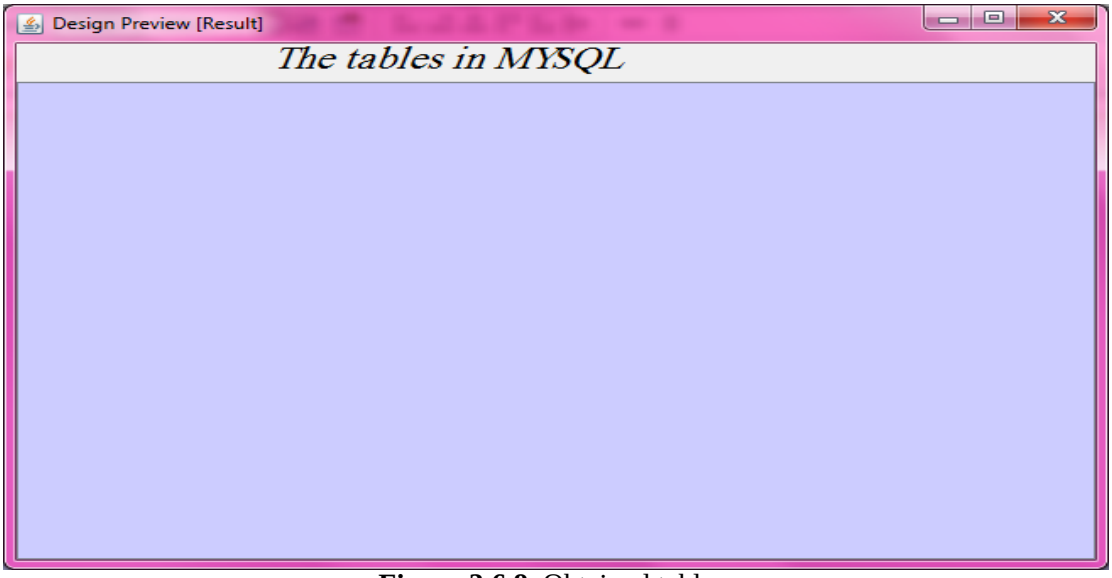

Figure 3.6.8 Obtained tables

7 Applicative example

Like we have mentioned in the second chapter, an XML document is composed from 2 parts. The first one is called DTD. The second part is the content of XML's document. In addition to the prolog at the beginning of the document.

7.1 XML document with DTD

7.1.1 The DTD

```
<!--The start of DTD part-->
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE family[
<!ELEMENT family(father,mother,daughter,son)>
<!ELEMENT father(#PCDATA)>
<!ATTLIST father idf ID #REQUIRED>
<!ATTLIST father name CDATA #IMPLIED>
<!ATTLIST father job CDATA #IMPLIED>
<!ATTLIST father adresse CDATA #FIXED>
<!ELEMENT mother(#PCDATA)>
<!ATTLIST mother idm ID #REQUIRED>
<!ATTLIST mother name CDATA #IMPLIED>
<!ATTLIST mother job CDATA #IMPLIED>
<!ATTLIST mother idf IDREF #REQUIRED>
<!ATTLIST mother adresse CDATA #FIXED>
<!ELEMENT daughter(#PCDATA)>
<!ATTLIST daughter idd ID #REQUIRED>
<!ATTLIST daughter name CDATA #IMPLIED>
<!ATTLIST daughter idf IDREF #REQUIRED>
<!ATTLIST daughter idm IDREF #REQUIRED>
<!ATTLIST daughter adresse CDATA #FIXED>
<!ELEMENT son(#PCDATA)>
<!ATTLIST son ids ID #REQUIRED>
<!ATTLIST son name CDATA #IMPLIED>
<!ATTLIST son idf IDREF #REQUIRED>
<!ATTLIST son idm IDREF #REQUIRED>
<!ATTLIST son idd IDREF #REQUIRED>
<!ATTLIST son adresse CDATA #FIXED> ]
<!--The end of DTD part-->
```

7.1.2 The content of XML

```
<!--The start of xml part-->
<family>
<father>
<idf>1</idf>
<name>ali</name>
<job>teacher</job>
<adresse>70-rue-Elmodjahidine</adresse>
</father>
<mother>
<idm>2</idm>
<name>rahil</name>
<job>dentist</job>
<idf>1</idf>
<adresse>70-rue-Elmodjahidine</adresse>
</mother>
<daughter>
<idd>3</idd>
<name>nassima</name>
<idf>1</idf>
<idm>2</idm>
<adresse>70-rue-Elmodjahidine</adresse>
</daughter>
<son>
<ids>3</ids>
<name>nassim</name>
<idf>1</idf>
<idm>2</idm>
<idd>3</idd>
<adresse>70-rue-Elmodjahidine</adresse>
</son>
</family>
<!--The end of  Xml document-->
```

In the following figures we open the precedent XML document with DTD . Then we fill the informations of MYSQL login , any errrors will cancel the transformation.

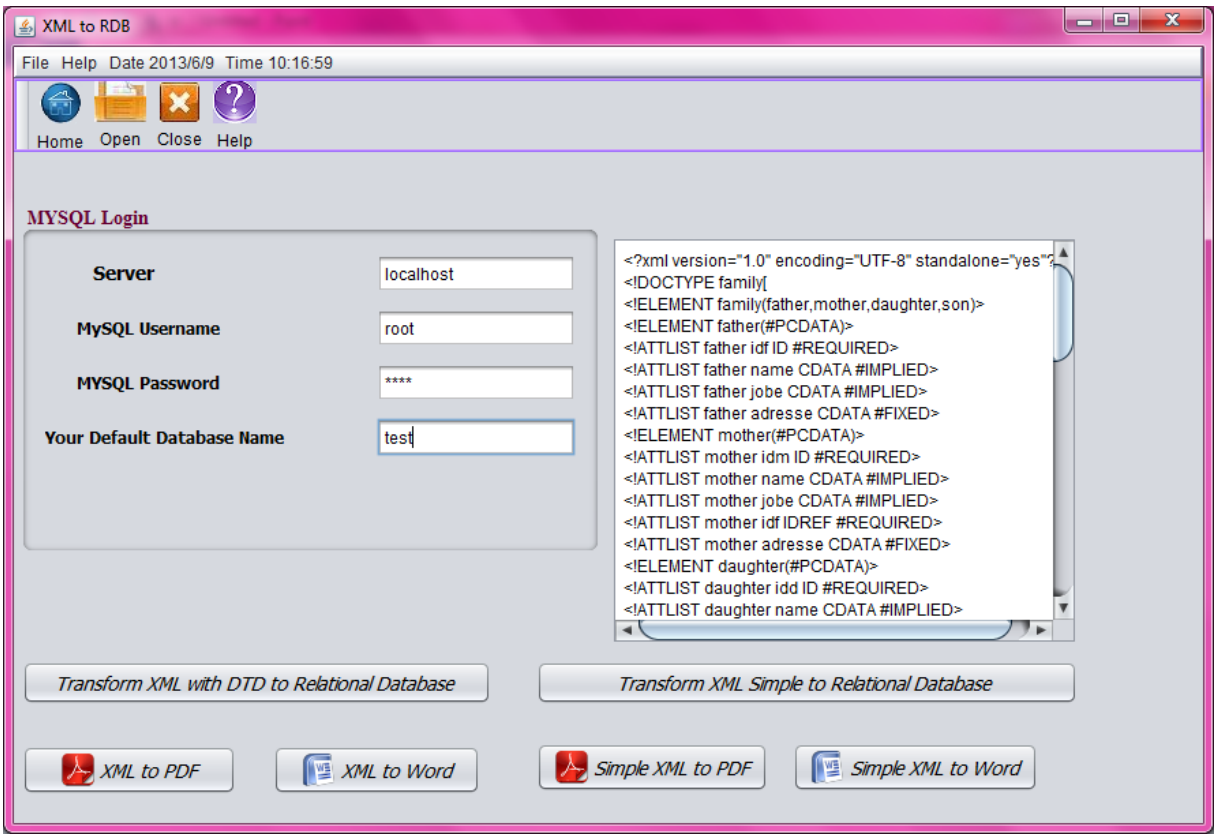

figure3.7.1 Filling MySQL information

7.2 Simple XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<family>
<father>
<idf>1</idf>
<name>ali</name>
<job>teacher</job>
<adresse>70-rue-Elmodjahidine</adresse>
</father>
<mother>
<idm>2</idm>
<name>rahil</name>
<job>dentist</job>
<idf>1</idf>
<adresse>70-rue-Elmodjahidine</adresse>
</mother>
<daughter>
<idd>3</idd>
<name>nassima</name>
<idf>1</idf>
<idm>2</idm>
<adresse>70-rue-Elmodjahidine</adresse>
</daughter>
<son>
<ids>3</ids>
<name>nassim</name>
<idf>1</idf>
<idm>2</idm>
<idd>3</idd>
<adresse>70-rue-Elmodjahidine</adresse>
</son>
</family>
```

The figure **3.7.2** is about the simple XML which we have treated. As the first type (with DTD) we must enter all the information about the sign in MySQL server (or MySQL Workbench).

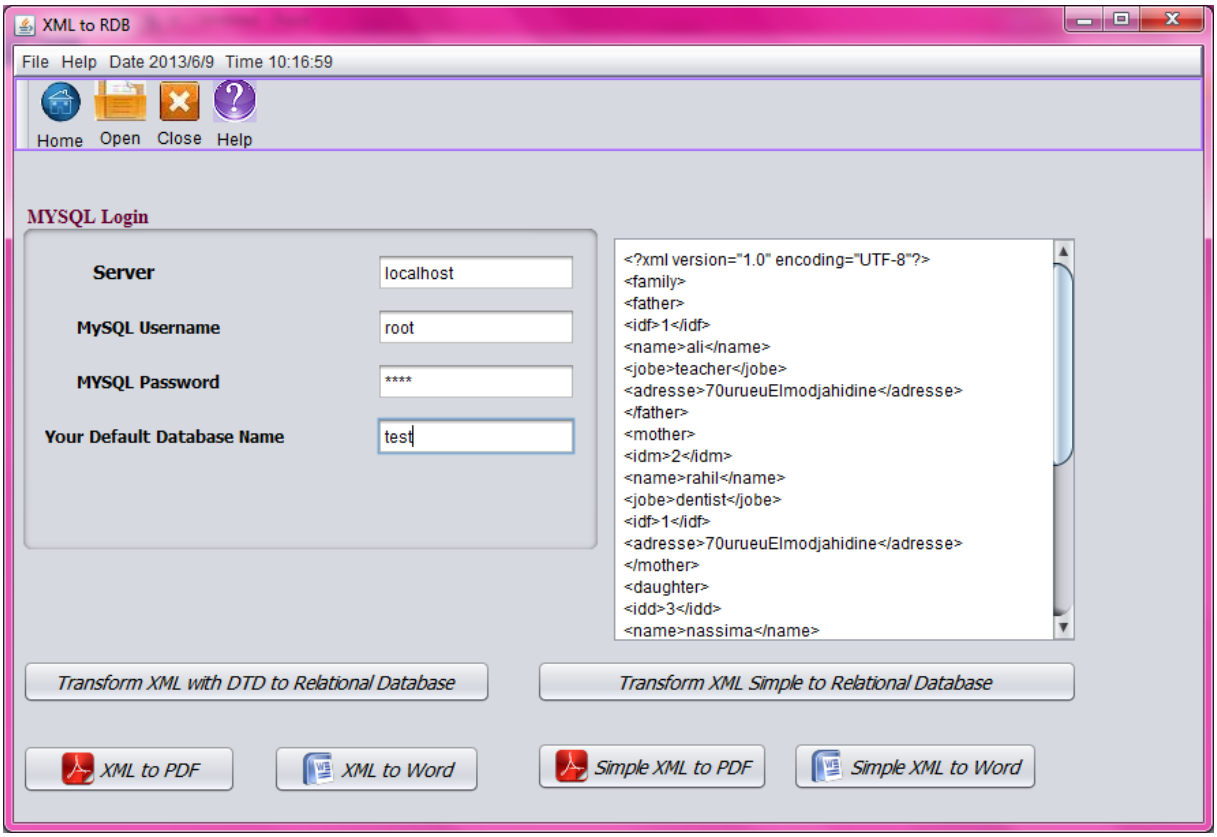

Figure 3.7.2 Filling MySQL information.

The results of generation process are the same for the two kinds of documents like in **Figure 3.7.3** and **Figure 3.7.4,** because we applied the same rules considering the content block.

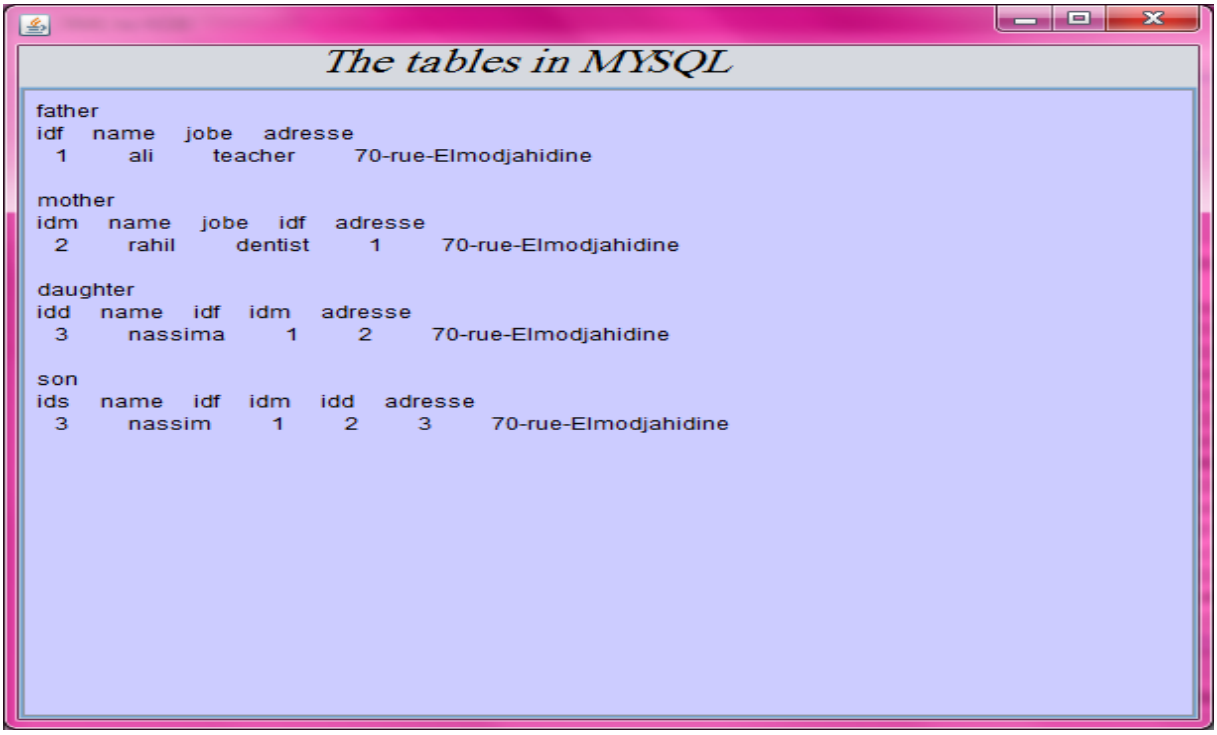

Figure 3.7.3 Obtained MySQL tables

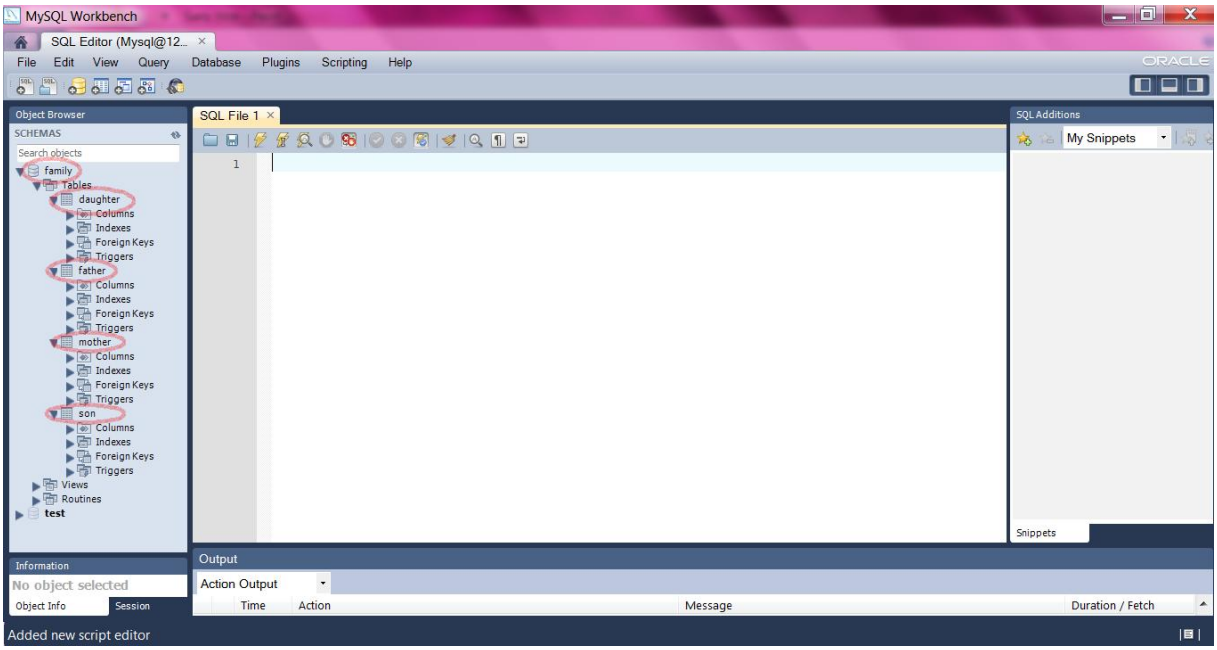

Figure 3.7.4 The target model in MYSQL Workbench

8 Conclusion

This chapter describes our implementation, including programming language used ,which is NETBEANS (for programming java) and (SGBD) MySQL Workbench (for creating relational database) . We have demonstrated the principle functionalities of our application by giving an example presenting results obtained from the two kinds of XML document.

The transformation from XML to relational database is very important technique in processing data; one common reason for transforming is that existing SQL applications might still require access to the data in relational format. For example, legacy applications, packaged business applications, or reporting software do not always understand XML and have fixed relational interfaces. Therefore you might sometimes find it useful to share all or some of the data values of an incoming XML document into rows and columns of relational tables.

In our work the first step was creating classes to analyze if the XML (with DTD or simple) document is valid, the second step is checking the prolog, and the tags. After that, checking SQL errors process must started. If no errors occurred, a relational database is generated from the XML document.

The developed application could just detecting errors, showing simple messages. As a future work we propose that those errors will be treated in order to help users having more ability when they use this application.

Besides this we have mentioned that we have used the DTD as grammar for XML document. As a perspective , our application can be ameliorated to manipulate XML schema in the place of DTD.

Bibliographies

Books

[1]Bill Kasdorf , XML Models for Books ,General Editor, The Columbia Guide to Digital Publishing

[2] Bill Trippe, Dale Waldt, Using XML and Databases, February 2008

[3]Casare Pautasso, XML Extensible Markup Language, university della Svissera italy 7.11.2007,

[4] D. Gasevic, D. Djuric, V. Devedzic and al., Model Driven Engineering and Ontology Development, Springer, May (2009).

[5] D.S. Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing, Wiley, (2003).

[6]Gilles, ARCHITECTING SOFTWARE SYSTEMS USING MODEL TRANSFORMATIONS AND ARCHITECTURAL FRAMEWORKS, Faculté des Sciences, de la Technologie et de la Communication, Université du Luxembourg Institut d'Informatique, Université de Namur (FUNDP).

[7] Joaquin Miller and Jishnu Mukerji, Model Driven Architecture(MDA), Architecture Board ORMSC1,April 18, 2001 .

[8] K. Czarnecki. Dept. Electrical and Computer Engineering, University of Waterloo200 University Ave., West Waterloo ON N2L 3G1, Canada czarnecki@acm.org

[9] Krzysztof Czarnecki and Simon Helsen,Classification of Model Transformation Approaches,University of Waterloo, Canada

[10]Matthias Biehl, Literature Study on Model Transformations .Institute of Technology,Stockholm, Sweden biehl@md.kth.se July 2010

[11] Patrick van Bommel,Transformation of knowledge, information and data , University of Nijmegen, The Netherlands.

Websites

[12] http://www.synonym.com/definitions/transformation/ (last seen 04/06/2013)

[13] http://drops.dagstuhl.de/volltexte/2005/11/pdf/04101.SWM2.Paper.pdf (last seen 04/06/2013)

[14] http://www.wisegeek.org/what-is-a-relational-database.htm# (last seen 04/06/2013)

[15] http://www.ehow.com/facts_7618553_difference-between-xml-database.html (last seen 04/06/2013)

[16] http://www.w3schools.com/xml/xml_dtd.asp (last seen 04/06/2013)

[17] http://faculty.madisoncollege.edu/schmidt/xml/schema.html (last seen 04/06/2013)

[18] http://www.w3schools.com/dtd/dtd_intro.asp (last seen 04/06/2013)

[19]http://www.coderanch.com/t/125567/XML/difference-PCDATA-CDATA (last seen 04/06/2013)

[20] http://www.freebookcentre.net/programming-books-download/Introduction-to-XML-for-Web-Developers-.html (last seen 04/06/2013)

[21] http://www.derekfountain.org/articles/dtd.pdf (last seen 04/06/2013)

Glossary

Glossary

[A]

- **AndroMDA:** is a code generation framework that follows the Model Driven Architecture (MDA) paradigm.
- **Atom:** The name Atom applies to a pair of related Web standards. The Atom Syndication Format is an XML language used for web feeds, while the Atom Publishing Protocol (AtomPub or APP) is a simple HTTP-based protocol for creating and updating web resources.
- **ArcStyler** : is one of the leading software development tools for Model Driven Architecture (MDA) It is a cross-platform, standards-compliant environment, fully implemented in Java, for the design, modeling, generation, deployment and management of high-quality, industrial strength applications of any size for architectures based on Java/J2EE and .NET as well as custom infrastructures and existing legacy platforms.
- **ATL:** ATLAS Transformation Language.

[B]

- **b+m Generator FrameWork :** The b+m Generator FrameWork from b+m Informatik AG is an open next-generation OMG MDA conform tool.

[C]

- **CERN : E**uropean **O**rganization for **N**uclear **R**esearch is an international organization whose purpose is to operate the world's largest particle physics laboratory. Established in 1954
- **CIM:** Computation Independent Model.

[D]

- **DBMS:** Database management systems are specially designed applications that interact with the user, other applications, and the database itself to capture and analyze data.
- **DTD:** Document Type Definitions.

[E]

- **EMF**: Eclipse Modeling Framework.

- **ETL: The** Epsilon Transformation Language.

[G]

- **GenGen:** Genetic Genomics Analysis of Complex Data is a suite of free software tools to facilitate the analysis of high-throughput genomics data sets.

[H]

- **HTML:** HyperText Markup Language; is the main markup language for creating web pages and other information that can be displayed in a web browser.

[M]

- **MDA:** Model Driven Architecture.
- **MDD:** Model Driven Development.
- **MDE:** Model-driven engineering.
- **MOF:** The Meta-Object Facility; is an Object Management Group (OMG) standard for model-driven engineering.

[O]

- **OAW: Open** Architecture Ware.
- **OMG:** Object Management Group, the consortium responsible for CORBA (Common Object Request Broker Architecture), Unified Modeling Language (UML), and Model-Driven Architecture (MDA) .
- **OWL**: Web Ontology Language is a family of knowledge representation languages for authoring ontologies.

  [P]
- **PDM: The** Platform Description Model.
- **PIM: Platform** Independent model.
- **Prolog** : a general purpose **logic programming** language associated with artificial intelligence and computational linguistics.
- **PSM: Platform** Specific Model.

[Q]

- **QVT** :(Query/View/Transformation) is a standard set of languages for model transformation defined by the Object Management Group.

[R]

- **RDBMS**: A relational database management system is a database management system (DBMS) that is based on the relational model .

[S]

- **SiTra:** is a simple Java library for supporting a programming approach to writing transformations and consists of two interfaces and a class that implements a transformation algorithm.
- **SGML:** the Standard Generalized Markup Language; is an ISO-standard technology for defining generalized markup languages for documents.
- **SQL**: Structured Query Language is a special-purpose programming language designed for managing data held in a relational database management system.
- **Sun :** was a company that sold computers, computer components, computer software, and information technology services and that created the Java programming language, and the Network File System (NFS).
- **SWRL**: Semantic Web Rule Language is a proposal for a Semantic Web rules language, combining sublanguages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language (Unary/Binary Datalog).

[T]

- **TCS:** Textual Concrete Syntax is an Eclipse/GMT component that enables the specification of textual concrete syntaxes for Domain-Specific Languages (DSLs) by attaching syntactic information to metamodels. With TCS, it is possible to parse (text-to-model) and pretty-print (model-to-text) DSL sentences. Moreover, TCS provides an Eclipse editor, which features: syntax highlighting, an outline, hyperlinks, and hovers for every DSL which syntax is represented in TCS.

- **TGG:** Triple Graph Grammars.
- **Tefkat:** is a Model Transformation Language and a model transformation engine. The language is based on F-logic and the theory of stratified logic programs. The engine is an Eclipse plug-in for the Eclipse Modeling Framework .
- **TXL:** is a unique programming language specifically designed to support computer software analysis and source transformation tasks.

[U]

- **UML**: Unified Modeling Language is a standardized, general-purpose modeling language in the field of software engineering.
- **UMLX:** is a concrete graphical syntax to complement the OMG QVT model transformation language.

[V]

- **VIATRA** (VIsual Automated model TRAnsformations) framework is the core of a transformation-based verification and validation environment for improving the quality of systems designed using the Unified Modeling Language by automatically checking consistency, completeness, and dependability requirements.

[W]

- **W3C**: The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web (abbreviated WWW or W3).

[X]

- **XML**: Extensible Markup Language.
- **XSLT**: Extensible Style sheet Language Transformations is a language for transforming XML documents into other XML documents, or other objects such as HTML for web pages, plain text or into XSL Formatting Objects which can then be converted to PDF, PostScript and PNG.
- **Xtext:** is an open sourse frame work for developing programming languages and specific languages (DSLs). Unlike standard parser generators, Xtext not only generates a parser, but also a class model for the Abstract Syntax Tree and a fully featured, customizable Eclipse-based IDE.